Available online at www.sciencedirect.com

ScienceDirect

Procedia - Social and Behavioral Sciences 96 (2013) 844 – 855

Procedia
Social and Behavioral Sciences

13th COTA International Conference of Transportation Professionals (CICTP 2013)

Optimization on Retrieving Containers Based on Multi-phase Hybrid Dynamic Programming

Zhan BIAN*, Zhi-hong JIN

Transport Management College, Dalian Maritime University, Dalian, 116026, P R China

Abstract

Retrieving export containers from a container yard is an important part of the ship-loading process. This paper presents a three-phase hybrid algorithm to solve for an optimized working plan for a gantry crane to retrieve all the containers from a given yard according to a given order. The optimization goal is to minimize the number of container movements, as well as the crane's working time. After generating an initial feasible retrieving sequence with heuristic rules, phase two obtains several alternative retrieving sequences through various methods. With a network, phase three constructs a shortest path problem and derives the optimal sequence by dynamic programming. Numerical testing results show that the algorithm is able to solve instances with more than 2000 containers, which is within the range of real-world applications. Moreover, the number of movements approaches the lower bound in most cases, and the resulting retrieving sequence is efficient.

Keywords: containers; retrieving sequence; heuristic rules; dynamic programming; three-phase hybrid algorithm

1. Introduction

Over the past 20 years, container terminals have witnessed the increased world-wide flow of containers and more popular larger-sized container vessels. The competition among terminals has become prominent which makes the efficiency of port operation an important factor in succeeding in the fierce competition. Of all of the popular service performance measures, vessel turnaround time, which is the average time that a vessel stays in a terminal, is the most important. Port operations can be generally divided into two parts: the discharging operation during which containers are unloaded from containerships, and the loading operation during which containers are loaded onto ships. In most container terminals, a large portion of the turnaround time of a vessel is consumed by the two processes. In this paper, we study the problem of retrieving containers from a yard in a given sequence,

* Corresponding author. Tel.: +86-15940959040.
E-mail address: bianzhan1990@163.com

doi:10.1016/j.sbspro.2013.08.096

which is an important part of the ship-loading process. Most yards stack up containers to utilize more and more precious space. Unlike many usual storage systems that are capable of providing random access to all stored items, only those located at the top are directly accessible to the yard cranes. In addition, containers have to be loaded onto ships according to the stowage plan, which specifies the location of each container on the ship, and thus largely determines the order the containers are to be loaded onto the vessel. Extra movements, that waste time and money, occur when a container is due to be retrieved from the yard but is buried beneath other ones. One potential way to reduce relocations and loading time is to pre-marshal the export containers before loading starts. Lee and Hsu (2007) focused on the container pre-marshaling problem for a single bay, and developed an integer programming model that yields a step-by-step working plan for the crane. Working in a similar direction, Lee and Chao (2009) solved much larger instances with a neighborhood search approach. However, all these models focus on yard pre-marshaling, and are different from the problem addressed in the current work.

The literature related to optimizing the retrieval process in a yard is not extensive. It was first proposed by Chung et al. (1988), then Kim and Kim (1997, 1999) formulated a mixed integer programming model for the routing problem of a crane loading export containers out of the stack area onto waiting yard trucks, but the scope of this model is limited to the routing of container carriers. Kim and Hong (2006) used a branch-and-bound approach, as well as a simple and effective heuristic, to determine how containers in a single bay can be retrieved with the least number of movements. The largest example presented in their paper has 30 containers, which is significantly smaller than the typical number of containers in practice. Also attempting to minimize re-handles, Lee and Lee (2010) proposed a three-phase heuristic and integer programming to solve for an optimized working plan for a crane.

In this paper we develop a hybrid algorithm for the container retrieval problem. Given the initial layout of a yard with multiple bays, the algorithm obtains a movement sequence for the crane to retrieve all the containers in a specified order. The optimization goal is to minimize the total number of container movements, as well as the overall working time. The algorithm is able to solve instances with more than 2000 containers in a few hours, which fits well into the time window between the arrival of a ship and the starting of the loading process. This performance brings the algorithm within the range of practical applications.

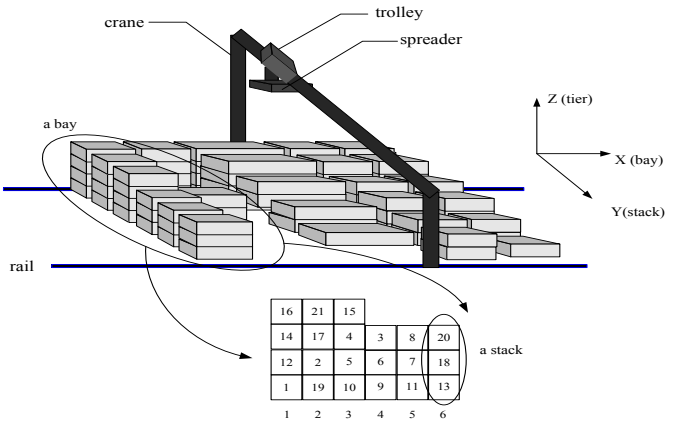

Fig. 1. Illusion of a rail mounted gantry crane retrieving containers among blocks.

The operating strategy of any container yard has to be developed according to the equipment used in that particular location. In this research, we focus on yards that use rail mounted gantry cranes (RMGC) as their major container handling equipment. As illustrated in Fig. 1., the crane is composed of a gantry that moves in the bay dimension, a trolley that moves in the stack dimension on top of the gantry, and a spreader that moves in the tier

dimension. When an RMGC lifts a container from the yard, the crane first positions itself at the bay where the target container resides, positions its trolley over the target stack, lowers its spreader to hold onto the container, and then lifts it up. After lifting the container, the RMGC can lower the container onto a truck waiting at one end of the crane, or place the container on the top of another stack. We assume the RMGC is capable of moving containers between different bays, as newer models commonly do. We also assume that there is only one RMGC in the working area, and thus collision between cranes is not a problem.

This paper is divided into five sections. Following this introduction, the second section will define the container retrieval problem. The mathematical methods will be developed in detail in the third section, followed by computational examples and analysis in section four. Finally, conclusions and future research will be discussed in the fifth section.

2. The container retrieval problem

Given an initial layout of a yard, the container retrieval problem yields a movement sequence that retrieves all the containers from the yard, one at a time in a specified order, such that the number of container movements as well as working time is minimized.

The basic assumptions used in this research are listed below:

1. Containers of different dimensions are stored in different stacks. The vast majority of containers are either 20 or 40-feet long. In principle, containers of different lengths can be mixed together in one single stack. For example, one 40-feet container can rest on two 20-feet containers (but not the other way round). However, doing this complicates yard operations and is avoided whenever possible in practice. To simplify notations and explanations, we will assume that all the containers are of the same length.

2. Only one ship is presumed to be present. Although it is possible for large ports to have two or more ships being loaded at the same time with containers taken from the same yard, served by the same RMGC, we do not consider this scenario.

3. The loading order of the containers is known. In practice, the loading sequence of a ship is determined well before loading starts (more than 6 h in most cases).Throughout the paper, the containers are numbered with consecutive integers starting from 1, and those with smaller numbers have to be retrieved earlier, shown as Fig. 1..

Fig. 2(a) shows a bay with three stacks, where all six containers can be retrieved without additional re-handles. In the bay shown in Fig. 2(b), containers 1 can be retrieved directly, but containers 4 and 6 have to be relocated to other stacks before containers 2 and 3 can be retrieved. Therefore, a lower bound for the number of relocation movements needed to retrieve all six containers in this stack is two, and the number of total movements is at least eight (the total number of containers plus the minimum number of relocations). A lower bound of the total number of movements for the entire yard can be easily estimated by summing up the minimum number of movements for all the stacks in the yard.

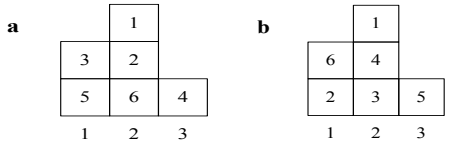

Fig. 2. Bays that are retrieved with different numbers of movements.

Similarly, we number the stacks with consecutive integers starting from 1. For example, if there are 16 stacks in each bay, then the stacks in the first bay are numbered from 1 to 16, the stacks in the next bay are numbered from 17 to 32, and so on. Stack 0 represents the truck which parks at one end of the crane to carry the retrieved container away. In this research we use a triplet consisted of the container identification number, the originating

stack, and the operation type to represent a container movement. For example, the movement (x, m, U) represents a lifting operation of container x from stack m; the movement (x, n, D) represents loading container x to stack n. A movement sequence is an ordered set of container movements and will be considered infeasible when one or more following conflicts occur:

1. The container is not in the stack while the crane executes a lifting operation.
2. There are misoverlays when the crane lifts the target container.
3. The spreader is already fully loaded when lifting operation begins.
4. The stack has reached its highest height when a container is to be loaded into.
5. The spreader does not hold the container or the spreader is empty when executing a loading operation.
6. Containers with bigger numbers leave the yard earlier than those with smaller numbers.

A feasible movement sequence contains a number of movements arranged in a certain order that, when executed, retrieves all the containers in the preferred order without encountering conflicts. (x, m, U) along with (x, n, D) form a complete movement pair. Thus we define the number of pairs as the sequence length. The quality of a feasible movement sequence can be determined by two aspects. One is the sequence length, and the other is the total time the crane needs to execute it. The former can be determined by simple counting, and the latter can be accurately estimated from the performance data of the crane, which specifies the time needed for the crane to reposition itself between movements, and the time for the crane to perform the movements themselves. The optimization goal of the algorithm proposed in this research is to derive an optimal movement sequence with the minimum working time.

3. The three-phase hybrid algorithm

The hybrid algorithm consists of three phases executed one after the other, namely the initial phase that generates a feasible retrieving sequence with heuristic rules, the second phase that obtains several alternative retrieving sequences through various ways and the third phase that derives an optimal sequence by dynamic programming. We next introduce the three phases in more detail.

3.1. The initial phase

The task of the initial phase of the heuristic is to develop a feasible retrieving sequence that enables the crane to retrieve all the containers in the yard without encountering conflicts. The following notations will be used to describe the heuristic:

Notations

N	The number of containers in the initial layout.
S	The number of stacks in the layout.
H	The maximum height of stacks.
a_n	The container with the serial number n, $n \in (1,2,...,N)$.
$s(a_n)$	The serial number of the stack where a_n stays, $s(a_n) \in (1,2,...,S)$.
$top(s)$	The serial number of the container on the top of stack s, if there is no container in stack s, then $top(s) = N+1$.

$b_n = top\left(s\left(a_n\right)\right)$ The container on the top of stack $s\left(a_n\right)$.

$\min\left(s\right)$ The minimum serial number of containers in stack s.

$U = \left\{s \mid s \in S, \min\left(s\right) > b_n\right\}$ The set of stacks satisfying the constraint $\min\left(s\right) > b_n$.

$\alpha = \underset{i \in S \setminus \left\{s\left(a_n\right)\right\}}{\arg\max}\left(\min\left(i\right)\right)$ Define the stack (any stack except for stack $s\left(a_n\right)$) with maximum $\min\left(i\right)$ as stack α.

$\beta = \underset{i \in U}{\arg\min}\left(\min\left(i\right)\right)$ Define the stack (of the set U) with the minimum $\min\left(i\right)$ as stack β.

$E\left(s\right)$ The number of empty slots in stack, $E\left(s\right) \in \left\{1,2,...,\mathrm{H}\right\}$.

$OBT\left(a_n\right)$ The set of blocked containers to container a_n.

ST The set of blocked containers satisfying the following constraints-its serial number is larger than b_n, on the top of stacks (except for stack s), $\min\left(s\right) - 6 < k < \min\left(s\right)$ (k represents the serial number of a blocked container).

st The container of the set ST with maximum serial number.

In this phase, the heuristic attempts to retrieve all the containers in the required order. If the target container is readily available, it is retrieved and loaded onto the truck immediately. Otherwise, the containers that block the target container are moved to certain stacks by heuristic rules. Fig. 3. illustrates the decision tree of retrieving sequence.

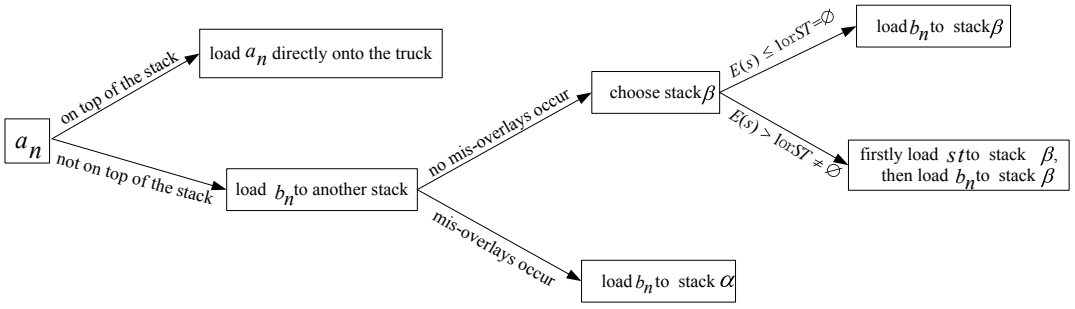

Fig. 3. The decision tree of retrieving sequence.

As is shown in Fig. 3., two different circumstances occur when choosing a temporary stack s for container b_n:

1. $\min(s) > b_n$, i.e., there is no more misoverlays after container b_n loaded to stack s. If several stacks satisfy the condition, choose $\beta = \arg\min(\min(i))$. Thus, the serial number of b_n will be close to the minimum serial number of containers in stack β, so as to reduce the probability for occupying the slot for the container of which serial number is bigger than that of b_n. If stack β with several empty slots exists, firstly check whether there are any ST to utilize the slots, then load ST to stack β according to descending order till b_n is in stack β or stack β reaches its maximum height.

2. $\min(s) < b_n$, i.e., more misoverlays will occur after container b_n loaded to any stack. Therefore, the stack which has the minimal impact on subsequent operation after b_n loaded will be chosen. It is obvious that the bigger for the serial number of overlay container by container b_n, the later the next movement of container b_n happens. So $\alpha = \arg\max(\min(i))$ is chosen as the temporary stack for container b_n.

The heuristic rules can be illustrated with the bay shown in Fig. 4..

The initial retrieving sequence obtained is (1,1,U), (1,0,D), (7,3,U), (7,6,D), (6,2,U), (6,6,D), (2,2,U), (2,0,D), (11,5,U), (11,4,D), (3,5,U), (3,0,D), (4,3,U), (4,0,D), (5,2,U), (5,0,D), (6,6,U), (6,0,D), (7,6,U), (7,0,D), (8,6,U), (8,0,D), (12,1,U), (12,2,D), (9,1,U), (9,0,D), (11,4,U), (11,2,D), (10,4,U), (10,0,D), (11,2,U), (11,0,D), (12,2,U), (12,0,D). It is easy to see that in the initial retrieving sequence, every container with misoverlays (e.g. container 2) is moved at least twice (once for moving out of the way of the blocked container, and again for retrieval), and every no-blocked container is moved only once (e.g. container 1).

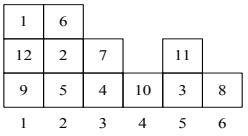

Fig. 4. Illustration of the initial yard plan.

3.2. The second phase

The second phase takes the initial feasible sequence as the input, and attempts to find more alternative movements while maintaining feasibility. Basic ideas to generate alternative sequences are as follows:

1. Reverse two pairs of container movements. If there are four different stacks involved in two pairs of container retrieving movements, we can change the order of the two pairs.

2. The initial stack is replaced by the alternative stack. For instance, there is a sequence composed of (x, m, U), (x, n, D),…, (x, n, U), (x, s, D), which does not contain other containers besides container x between every two movements. If stack t is completely not used in the sequence, and no more misoverlays occur after loading container x into stack t, the sequence can be replaced by (x, m, U), (x, t, D),…, (x, t, U), (x, s, D).

3. Bring movements forward. This method goes for a container with more than two pairs of movements. Assuming that container x owns two pairs of movements (x, m, U), (x, n, D),…, (x, n, U), (x, s, D), but the whole sequence is (x, m, U), (x, n, D),…, (y, p, U), (y, q, D), (x, n, U), (x, s, D). As stacks use in the pair (x, n, U), (x, s, D) are different from that of (y, p, U), (y, q, D), (x, n, U), (x, s, D) can be moved to the front of (y, p, U), (y, q, D), and brought forward till meet a pair with the same stacks or a previous movement pair for container x (i.e. (x, m, U), (x, n, D)).

4. Bring movements backward. The method is almost the same as the idea 3, but of opposite searching direction.

5. Bring a pair forward. This method goes for a container with only one pair of movements which is aimed at loading the container onto a truck. Assuming that container x only owns one pair of movements (x, k, U), (x, 0,

D), but movements in front of these do not use the same stack. Besides, containers with smaller serial number have been retrieved out of the yard. Based on the above conditions, (x, k, U), (x, 0, D) can be moved forward.

6. Bring a pair backward. The method is almost the same as the idea 5, but of opposite searching direction.

The ideas can be applied to generate alternative sequences for the initial retrieving sequence mentioned in 3.1. According to the idea 2, container 6 can be retrieved from stack 2 and loaded to stack 1. Thus, an alternative sequence can be indicated as(6,2,U), (6,1,D), (2,2,U), (2,0,D), (11,5,U), (11,4,D), (3,5,U), (3,0,D), (4,3,U), (4,0,D), (5,2,U), (5,0,D), (6,1,U), (6,0,D). According to the idea 1, we can reverse the order of the pairs (8,6,U), (8,0,D) and (12,1,U), (12,2,D). Therefore, another alternative sequence can be represented as (12,1,U), (12,2,D), (8,6,U), (8,0,D).

3.3. The third phase

Phase three aims to obtain the optimal sequence by the following steps: firstly, build a network without loops for all retrieving movements to formulate a shortest path problem, and then use dynamic programming to solve the problem.

Numerous alternative sequences can be derived though the six ideas above. We build a shortest path problem based on the network of which vertexes and edges represent the storage condition and time-consumption of moving and retrieving operation respectively. Fig. 5. and Fig. 6. are networks of the initial retrieving path and the alternative path with two alternative retrieving sequences (mentioned in 3.2) respectively. Vertex ① illustrates the initial storage condition (shown as Fig. 4.) and vertex ⑱ illustrates the final yard layout after all retrieving operations. Parameters used in dynamic programming which is proposed to solve the shortest path problem are as follows:

Parameters

N The number of containers in the initial layout.

m The number of bays in the initial layout.

n The number of stacks in a bay.

H The maximum height of stacks.

k The total number of stages, $k \in (1,2,...,\mathrm{N})$.

d_i^k The state of stack i after container k is retrieved, $k \in (1,2,...,\mathrm{N})$, $i \in (1,2,...,m \times n)$.

s_j^k The state of bay j after container k is retrieved, $s_j^k = \left\{ d^k_{(j\text{-}1)\times n+1}, d^k_{(j\text{-}1)\times n+2}, ..., d^k_{j\times n} \right\}$, $k \in (1,2,...,\mathrm{N})$, $j \in (1,2,...,m)$.

S^k The state of the yard after container k is retrieved, $S^k = \left\{ s_1^k, s_2^k, ..., s_m^k \right\}$, $k \in (1,2,...,\mathrm{N})$.

a^k All the movements taking place while retrieving container k, $k \in (1,2,...,\mathrm{N})$.

$time\left(a^k \middle| S^{k\text{-}1}\right)$ The time-consumption of moving and retrieving operations while conducting a^k under the state of S^{k-1}, $k \in (1,2,...,\mathrm{N})$.

$T\left(S^k\right)$ The minimal time-consumption of moving and retrieving operations to retrieve the rest of $N-k$ containers under the state of S^k, $k \in (1,2,...,\mathrm{N})$.

The problem can be described as: $T\left(S^0\right)=\min\left\{time\left(a^1 \middle| S^0\right)+T\left(S^1\right)\right\}$, where S^0 turns into S^1 via the retrieval decision $u_1\left(S^0\right)$ and the movement a^1, i.e., $S^0 \xrightarrow[a^1]{} S^1$. By parity of reasoning, the minimal time consumed by the whole process can be represented as: $T\left(S^0\right)=\min\left\{\sum time\left(a^p \middle| S^{p-1}\right)+T\left(S^k\right)\right\}$, where $S^{p-1} \xrightarrow{a^p} S^p$, $p=1,2,...,\mathrm{k}$. And the optimal retrieval decisions are $u_1\left(S^0\right),u_2\left(S^1\right),...,u_k\left(S^{k-1}\right)$. The second and the third phase are iterative, and will not terminate until consecutive iterations cannot minimize the RMGC working time.

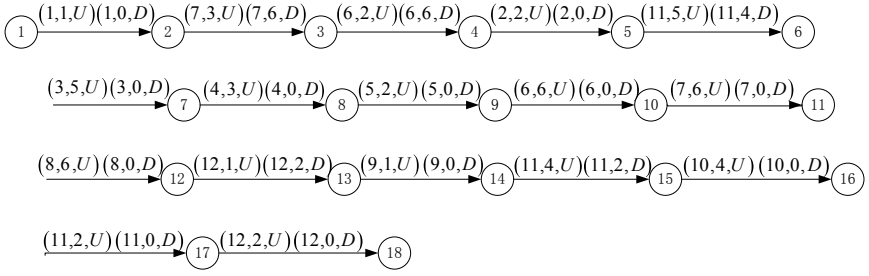

Fig. 5. Illustration of the initial retrieving path.

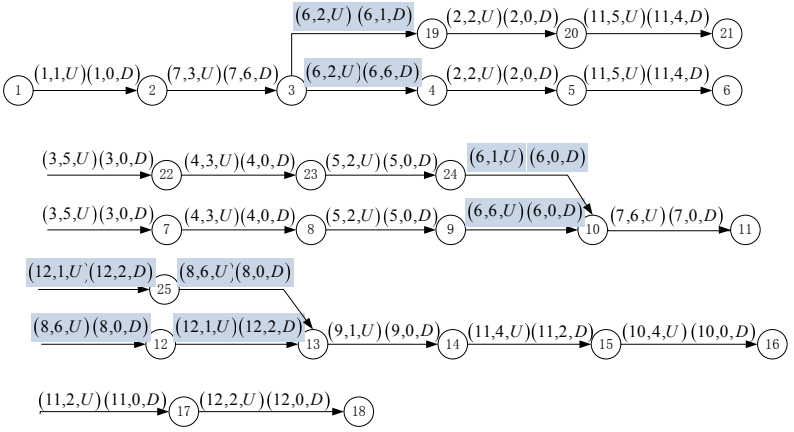

Fig. 6. Illustration of the alternative retrieving path.

4. Computational results

In this section, we provide computational examples to demonstrate the performance of the algorithm developed in this paper. The proposed algorithm has been implemented in Microsoft Visual C++ and run on a personal computer which has a Core I5 CPU running at 2.50GHz and with 4.0GB memory.

In all the cases, the containers are generated, and randomly placed in the yard, subject to pre- determined number of bays, stacks per bay, and maximum stack height. Some preference settings are shown as follows (the same as Lee and Lee (2010)): the speed of the gantry is 3.5 s per bay, the speed of the trolley is 1.2 s per container width, the acceleration and deceleration time loss combined for the gantry is 40 s, while the pick-up and place-down time combined for the spreader is 30 s.

Example 1 presents the case shown as Fig. 4. in section 3.1. It is a small instance with only 1 bay containing 6 stacks and a maximum height of 6. The estimated lower bound for the number of movements to retrieve all 12 containers is 16. However, the number of movements in sequence is 17 due to one relocation operation of container 11. The instance was solved in 0.936 s by the proposed algorithm with 502 iterations while in less than 0.1s at the initial phase with only 1 iteration. The retrieval work in this case can be completed in 1670.19 s by an RMGC via the proposed algorithm while in 1698.99 s at the initial phase.

Next we compare the current algorithm with the one proposed by Lee and Lee (2010). Their heuristic has already presented 50 instances (shown as Table 1 of Lee and Lee (2010)) which are also used as experiments here. The result of the 50 instances is shown in Table 1. The current algorithm resulted in fewer movements in all 50 instances, and the CPU time is significantly lower than Lee & Lee (2010), which makes the current algorithm of much more practical significance. Moreover, the RMGC working time is also optimized by the current algorithm.

Table 1. Comparison of the result with the heuristic by Lee and Lee (2010).

ID	L	M		CPU		T		T/M	
		the current algorithm	Lee & Lee (2010)	the current algorithm	Lee & Lee (2010)	the current algorithm	Lee & Lee (2010)	the current algorithm	Lee & Lee (2010)
R011606_0070_001	100	107	118	2.28	6304.28	9672.6	10832.2	90.4	91.8
R011606_0070_002	104	110	117	3.77	11081.03	10406.6	10840.2	88.9	92.7
R011606_0070_003	104	104	110	1.87	5501.92	9913.2	10326.2	90.1	93.9
R011606_0070_004	108	108	158	3.07	9026.42	13270.3	13823.2	84.0	87.5
R011606_0070_005	106	112	124	2.26	9107.97	10926.8	11405.8	88.1	92.0
R021606_0140_001	208	208	228	9.96	21579.89	20857.0	21771.4	91.5	95.5
R021606_0140_002	197	197	224	9.46	21582.05	20535.1	21435.4	91.7	95.7
R021606_0140_003	211	223	247	9.53	21581.51	22232.9	23207.6	90.0	94.0
R021606_0140_004	219	219	235	13.27	21565.92	21408.9	22347.5	91.1	95.1
R021606_0140_005	210	210	217	10.79	21588.39	20335.3	20985.9	93.7	96.7
R041606_0280_001	439	439	502	173.84	21493.97	44119.5	49074.6	87.9	97.8
R041606_0280_002	423	423	450	188.75	21524.91	43720.2	45447.2	97.2	101.0
R041606_0280_003	415	419	450	179.30	21515.48	43455.6	45172.1	96.6	100.4
R041606_0280_004	426	426	430	170.27	21544.53	42449.8	44080.8	98.7	102.5
R041606_0280_005	431	431	439	188.87	21531.67	42850.8	44543.4	97.6	101.5

R061606_0430_001	660	660	765	189.53	21453.19	66462.0	78282.4	86.9	102.3
R061606_0430_002	654	670	695	200.02	21401.70	71199.1	74011.5	102.4	106.5
R061606_0430_003	656	656	698	199.39	21334.25	70763.4	73558.6	101.4	105.4
R061606_0430_004	648	648	699	211.45	21356.20	71224.7	74038.2	101.9	105.9
R061606_0430_005	660	660	701	198.59	21248.64	71553.0	74534.4	102.1	106.3
R081606_0570_001	869	869	924	206.64	21283.80	99003.5	103128.6	107.1	111.6
R081606_0570_002	874	874	930	196.59	21035.64	99657.6	103810.0	107.2	111.6
R081606_0570_003	891	891	981	192.95	21224.41	102785.0	107515.7	104.8	109.6
R081606_0570_004	871	871	952	197.35	21116.97	100663.9	105297.0	105.7	110.6
R081606_0570_005	873	873	940	199.48	21344.01	100401.0	105022.0	106.8	111.7
R101606_0720_001	1107	1107	1163	285.75	20753.88	147986.0	173205.5	127.2	148.9
R101606_0720_002	1085	1085	1132	261.39	20911.26	127460.8	133327.2	112.6	117.8
R101606_0720_003	1102	1102	1225	263.67	21093.99	134731.9	140933.0	110.0	115.0
R101606_0720_004	1081	1100	1168	276.07	20705.25	131173.5	137210.8	112.3	117.5
R101606_0720_005	1085	1085	1158	334.83	20759.42	129564.4	135527.6	111.9	117.0
R011608_0090_001	143	143	190	6.98	13268.67	16034.6	16772.6	84.4	88.3
R011608_0090_002	139	139	191	5.86	11134.63	16208.1	16883.4	84.9	88.4
R011608_0090_003	142	142	216	11.36	21583.13	18102.5	18856.8	83.8	87.3
R011608_0090_004	143	143	178	3.71	7042.38	15284.9	15921.8	85.9	89.4
R011608_0090_005	143	143	182	7.23	13738.00	15629.8	16281	85.9	89.5
R021608_0190_001	305	305	423	10.78	21552.80	36627.1	38101.6	86.6	90.1
R021608_0190_002	309	309	359	10.76	21527.84	32214.5	33511.4	89.7	93.3
R021608_0190_003	302	311	373	10.77	21539.86	33222.5	34560	89.1	92.7
R021608_0190_004	303	303	351	11.32	21498.59	31494.3	32762.2	89.7	93.3
R021608_0190_005	310	310	333	11.33	21519.48	30564.3	31794.8	91.8	95.5
R041608_0380_001	602	602	830	202.96	21310.55	74768.2	77778.2	90.1	93.7
R041608_0380_002	617	617	804	175.49	21058.70	74071.5	77053.5	92.1	95.8
R041608_0380_003	603	603	684	133.77	21269.45	64929.5	67634.9	94.9	98.9
R041608_0380_004	614	614	755	133.24	21317.80	69913.5	72932.9	92.6	96.6
R041608_0380_005	617	617	773	170.96	21198.89	71409.3	74493.3	92.4	96.4
R061608_0570_001	904	904	1143	186.08	20655.24	111125.5	115924.8	97.2	101.4
R061608_0570_002	897	897	1353	190.60	21156.92	124227.9	129593.1	91.8	95.8
R061608_0570_003	913	913	1139	204.66	20875.27	111150.1	115781.4	97.6	101.7
R061608_0570_004	902	910	1242	188.43	20915.81	117832.2	122614.2	94.9	98.7
R061608_0570_005	914	914	1333	179.03	20766.93	124113.7	129150.6	93.1	96.9

ID: Instance taken from Table 1 of Lee and Lee (2010).
L: Lower bound on the number of movements.
M: Number of movements in sequence.
CPU: CPU time, in seconds.
T: Crane working time.

T/M: Average time of each movement.

As the heuristic by Lee & Lee (2010) deals with no more than 10 bays, total 720 containers, in the last part of this session, we test the performance of the current algorithm with larger-sized cases, compared with the initial phase of the algorithm. Table 2 shows four more examples proposed in this part. For example, R101606_0816_001 presents a large instance with 10 bays, each containing 16 stacks. The maximum height of the stacks is 6, and there are 816 containers in the yard. Therefore, the space utilization rate is 85%. Table 3 shows the result of the current algorithm and the initial phase of it respectively. The initial phase took only one iteration to get the feasible retrieving sequence while the whole algorithm took much more iterations, as a result of the second phase and the third phase, to get the optimal sequence. Although the numbers of movements in two solutions are the same, the RMGC working time of the current algorithm is reduced dramatically compared with that of the initial phase. And even though CPU time of the current algorithm is about 30 times more than that of the initial phase, it is still within acceptable limits for randomly stacked yards.

Table 2. Additional numerical examples.

ID	Bay	Stack	Max Height	NO. of containers	Utilization
R101606_0816_001	10	16	6	816	85%
R101606_0864_001	10	16	6	864	90%
R200806_0720_001	20	8	6	720	75%
R301606_2160_001	30	16	6	2160	75%

Table 3. Comparison of the result with the initial phase.

ID	L	I		M		CPU		T		T/M	
		initial	current	initial	current	initial	current	initial	current	initial	current
R101606_0816_001	1279	1	617	1287	1287	8.45	255.94	193306.2	172392.4	150.2	133.9
R101606_0864_001	1359	1	621	1384	1384	8.79	286.05	207666.0	185619.5	150.0	134.1
R200806_0720_001	1121	1	602	1121	1121	8.21	260.59	182396.5	159173.2	162.7	142.0
R301606_2160_001	3321	1	898	3321	3321	322.73	8540.06	616130.2	515086	185.5	155.1

L, M, CPU, T, T/M: Same meaning as in Table 1.
I: Number of iterations in the phase or the whole algorithm.

5. Conclusions and future research

This study addressed the problem of retrieving containers from the container yard. A three-phase hybrid algorithm was proposed to solve the problem, which aims to minimize the number of container movements, as well as the RMGC's working time. The algorithm starts by generating an initial feasible retrieving sequence according to heuristic rules. Phase two attempts to find more alternative movements by six ideas. In the third phase, the algorithm uses dynamic programming based on a network to reduce the total working time the crane needs to complete the entire sequence without increasing the number of movements. The two later phases are both iterative, and terminate when a number of consecutive iterations cannot further improve the current solution. Numerical results show that the algorithm is able to solve instances of more than 2000 containers, and thus of practical use to the industry. Besides, the number of movements in the optimal solutions are close to their lower bounds.

In this paper, we investigated the container retrieving problem of one RMGC with a single spreader. However, cranes with multi-spreader are becoming increasingly popular in terminals. Moreover, there are more than two

cranes mounted on the same set of rails to work together for one task. The extension to the case of multiple types of RMGC is a promising topic for future research.

Acknowledgements

This research was partially funded by National Natural Science Foundation of China (No.71172108: Research on Synchronous Resources Scheduling in a Container Terminal based on Hybrid Flow Shop Arrangement), and by Dalian Science and technology project (No.2012A17GX125: Research on critical technologies for reduced containers relocation in a container terminal).

References

Lee Y-S, Hsu N-Y. (2007). An optimization model for the container pre-marshaling problem. *Computers and Operations Research*, 34(11): 3295-3313.

Lee Y-S, Chao S-L. (2009). A neighborhood search heuristic for pre-marshaling export containers. *European Journal of Operational Research*, 196(2): 468-475.

Chung Y G, Randhawa S U, McDowell E D. (1988). A simulation analysis for a transtainer-based container handling facility. *Computers & Industrial Engineering*, 14(2): 113-125.

Kim K Y, Kim K H. (1997). A routing algorithm for a transfer crane to load export containers onto a containership. *Computers and Industry Engineering*, (33): 673-676.

Kim K H, Kim K Y. (1999). An optimal routing algorithm for a transfer crane in port container terminals. *Transportation Science*, 33(1): 17-33.

Kim K H, Hong G-P. (2006). A heuristic rule for relocating blocks. *Computers and Operations Research*, 33(4): 940-954.

Lee Y-S, Lee Y-J. (2010). A heuristic for retrieving containers from a yard. *Computers and Operations Research*, 37: 1139-1147.